

Computing open-loop noncooperative solution in discrete dynamic games

Süheyla Özyildirim*

Department of Economics, Bilkent University, Bilkent, Ankara 06563

Abstract. The purpose of this paper is to introduce a new algorithm for the approximation of non-quadratic, non-linear open-loop Nash Cournot equilibrium in a difference game of fixed duration (multiperiod) and initial state. The algorithm based on adaptive search procedure called genetic algorithm has been used to optimize strategies for N -person dynamic games. Since genetic algorithms require little knowledge of the problem itself, computations based on these algorithms are very attractive to complex dynamic optimization problems. The empirical evidences are also provided to show the success of the algorithm developed. A typical example in US macroeconomic policy selection for 1933–1936 yields evidence of political inference in the economy.

Key words: Discrete Dynamic Game – Genetic Algorithm – Open-Loop Strategies

JEL-classification: C61; C73

1 Introduction

Economics and other social sciences are concerned with the dynamics arising from the interaction among different decision makers. Because the interactions do not always coincide, game-theoretic considerations become important. Game theory involves multi-person decision-making; it is dynamic if the order in which the decisions are made is important, and it is

* I am grateful to an anonymous referee for helpful comments. Remaining errors are my own.

noncooperative if each person involved pursues his own interests, which are partly conflicting with others (Başar and Oldser 1982).

One might argue that ideally all economic problems should be modeled as dynamic games, since each individual interacts constantly with others in a society. Many authors who think in this way have sought explicit solutions to dynamic and noncooperative games (eg. Kydland 1975, Pindyck 1977, De Bruyne 1979, Van der Ploeg 1982, Miller and Salmon 1985). However, a closer investigation of the available solutions reveals that most are actually only optimal subject to certain simplifying restrictions, which permit the derivation of *analytically tractable* decision rules. Because of their mathematical tractability and the possibility of obtaining an analytical solution, the linear quadratic games are well-suited to the purpose of deriving necessary and sufficient conditions for a noncooperative equilibrium.

In the last few decades, remarkable progress has been made in adapting control theory to economic problems, and the methods for computing solutions to the multi-person (N -person) problem are obtained by generalizing well known methods of optimal control theory (Starr and Ho 1969). In the traditional control theory, all the ingredients are singular in the sense that there is only one criterion, one central controller coordinating all control actions, and one information set available to the controller (Ho 1970). On the other hand, one can also argue that this viewpoint is unnecessarily narrow. Surely, we can all visualize situations or problems in which there is more than one criterion or performance measure, more than one intelligent controller operating with or without coordination from others, and finally, all the controllers may or may not have the same information set available to them. Generally, a problem of game theory is much more complex than its traditional one-player counterpart. In the simplest terms, the former has a full, at least two-dimensional matrix; the latter, a single-row matrix. In addition, there is no single satisfactory definition of optimality for these N -person problems. Depending on the applications, various types of solutions are relevant (eg. open-loop or closed-loop solutions).

In situations where analytical resources fail to cast light, computational simulations of a model can provide much needed *clues* to what constitutes the true behavior of the system in question. This approach has had considerable recent success in many areas of the physical and biological sciences and in mathematics itself. Indeed, whole areas of inquiry owe their existence to the careful examination of well conceived numerical computations (Bona and Santos 1994).¹

Economists are increasingly turning to numerical techniques for analyzing dynamic economic models (Judd 1992). While the progress has been substantial, the numerical techniques have tended to be, or at least have appeared to be, *problem specific*. Hence, this paper presents a new optimization algorithm as the numerical solution of noncooperative N -person nonzero sum difference games.²

¹ The numerical methods were originally developed by control theorists and their chief interest has been testing their capability of solving moderate sized economic problems (Kendrick and Taylor 1970).

² Difference games are dynamic games in discrete time.

The task of optimizing a complex system such as a dynamic game presents at least two levels of problems. First, a class of optimizing algorithms that are suitable for application to the system must be chosen. Second, various parameters of the optimization algorithm need to be tuned for efficiency (Grefenstette 1986). In this study, a class of adaptive search procedures called *Genetic Algorithm (GA)* has been designed to optimize complex systems such as noncooperative difference games. We will study open-loop Nash equilibrium solutions and leave problems of stability, uniqueness etc. aside. The challenge here is to introduce an efficient purpose algorithm to analyze more complex economic models.

The remainder of this paper is organized as follows: Open-loop noncooperative notations and solutions are described in Section 2. The Genetic Algorithm is introduced in Section 3 and the main numerical approximation algorithm is sketched out in Section 4. Numerical examples and their results are given in Section 5. Finally, a brief conclusion is given in Section 6.

2 Open-loop noncooperative solution in discrete dynamic games

The noncooperative solution implies that each player in the game maximizes his self-interest subject to his perception of the constraints on his decision variables. Noncooperative equilibrium solutions to nonzero sum discrete games were discussed in detail in Kydland (1975), Pindyck (1977), De Bruyne (1979), Brandsma and Hallett (1984) Karp and Calla (1983) and De Zeeuw and Van der Ploeg (1991) and also several references to literature can be found there.

The open-loop solution is a sequence of decisions for each time period, each of which depends on the initial state. The open-loop noncooperative solution presumes that at time 0, each player can make binding commitments about the actions he announces to undertake in the entire planning period.³ Each player in the game designs his optimal policy based on his own objectives at the beginning of the period, and sticks to that policy throughout the entire period.

In the original Kydland's description of N -player, nonzero difference game, the i th player chooses $u_{i1}, u_{i2}, \dots, u_{iT}$ trying to maximize (or minimize)

$$J_i = \sum_{t=0}^T L_i(x_t^i)$$

subject to

$$x_t^i = f_t^i(x_0, w_1^i, \dots, w_t^i, u_{i1}, \dots, u_{it}), \quad t = 1, \dots, T$$

³ Here, of course, the term "open-loop noncooperative" should be interpreted in a different context than in standard game theory. In the latter, noncooperative is used to imply the absence of binding commitments, whereas, in this paper, we use the same term to describe the fact that the agents have conflicting interests. The term "open-loop noncooperative" used in this paper, however, implies only that the players with conflicting interests have the ability to make binding commitments.

$$x_0, w_1^i, \dots, w_T^i \text{ given,}$$

where

$$w_t^i = (w_{1t}, \dots, w_{i-1,t}, w_{i+1,t}, \dots, w_{Nt})$$

is the player i 's expectation of the other players' decisions. (The possible inequality or equality constraints on the state and the control variables are omitted for simplicity.)

Player i has control over u_i only, but he has to consider what the other players do in the game. Thus, the open-loop solution of the optimization problem for player i in period t is the solution of the above problem, and solutions for each time period are N mappings

$$x_0, w_1^i, \dots, w_T^i \rightarrow u_{it}$$

where $i = 1, \dots, N; t = 1, \dots, T$.

The noncooperative solution will be

$$u_t = w_t = g_t^*(x_0), \quad t = 1, \dots, T$$

The quadratic approximation of the objective function with linear constraints has been extensively studied by Kydland (1975) and Pindyck (1977). In previous solutions of open-loop discrete dynamic games, the solution methodologies depend on assumptions that make the problem's mathematics reasonably tractable. Hence, in almost all studies in this area, each player arrives at his decision using the *same* econometric model (i.e., each has the same view of the way the world works), but has a *different set of objectives*: the possibility of two players having the same set of objectives but each exercising control based on decisions arrived at using different econometric models is not amenable to solution.

Such linear-quadratic dynamic games use the following solution procedure: There are N players, each with control vector \mathbf{u}_{it} where $i = 1, 2, \dots, N$ and $t = 1, \dots, T$. The evolution of the state \mathbf{x}_t is given by the linear difference equation

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{C}_t$$

where \mathbf{x}_t is K dimensional and \mathbf{u}_t is N dimensional. The objective of player i is to maximize (or minimize) $\mathbf{L}_i(\mathbf{x}_0)$ where

$$\mathbf{L}_i(\mathbf{x}_0) = \sum_{t=1}^T (\mathbf{r}'_{it}\mathbf{x}_t + \mathbf{x}'_t\mathbf{R}_{it}\mathbf{x}_t)$$

where $i = 1, 2, \dots, N$. The vectors and matrices $\mathbf{A}, \mathbf{B}_i, \mathbf{C}, \mathbf{r}_i, \mathbf{R}_i$ are given. They indicate the effect on the current state of the previous state (\mathbf{A}), current controls (\mathbf{B}_i) and the exogenous change (\mathbf{C}); \mathbf{r}_i and \mathbf{R}_i give the effect of the current state on player i 's single-period payoff. The inclusion of the controls in the state vector allows the function \mathbf{L}_i to depend on both the controls and the state.

Since $\mathbf{L}_i \neq \mathbf{L}_j$, the players have conflicting objectives. We seek a non-cooperative Nash solution to this game by finding a set of N strategies from which no player can unilaterally deviate without decreasing his payoff. Open-loop controls require that at the beginning of the game, each player

determines his entire trajectory of controls as a function of time. It is well known that when the objective function is quadratic and the equation of motion linear, optimal controls can be expressed as a linear function of state. Thus, it is not surprising that in the difference game, the equilibrium reaction functions are *linear* in the state:

$$\mathbf{u}_t = \mathbf{d}_t + \mathbf{E}_t \mathbf{x}_0$$

As in the control problems, \mathbf{d}_t and \mathbf{E}_t are independent of the state, but depend on the parameters of the problem.

Very little attention has been given to the development of *computational* techniques to solve N -person games and especially to find open-loop and closed-loop equilibrium controls (Pau 1975). Hence, the chief interest here is to describe a new technique for solving more general problems without making too many simplifications on the environment. The numerical algorithm described is used for the approximation of open-loop Nash equilibrium controls in a difference game of fixed duration and initial state.

3 Genetic algorithm

GAs are search algorithms based on the mechanics of natural selection and natural genetics.⁴ Even in large and complicated search spaces, given certain conditions on the problem domain, *GAs* tend to converge on solutions that are globally optimal or nearly so (Goldberg, 1989). A number of experimental studies have shown that *GAs* exhibit impressive efficiency in practice: While classical gradient search techniques are more efficient for problems which satisfy *tight* constraints (e.g. continuity, low dimensionality, unimodality etc.), Genetic Algorithms consistently outperform both gradient techniques and various forms of random search on more difficult (and more common) problems, such as optimization involving discontinuous, noisy, high dimensional and multimodal objective functions. A *GA* performs a multi-directional search maintaining a population of potential solutions. This population undergoes a simulated evolution: at each generation the relatively “good” solutions reproduce, while relatively “bad” solutions die. Hence, it is an iterative procedure which maintains a constant size population of candidate solutions or structures. During each iteration step, called a generation, the structures in the current population are evaluated, and on the basis of those evaluations, a new population of candidate solutions are formed. Structures are usually coded as strings of characters drawn from some finite alphabet (often the binary alphabet; 0,1). Also, *GAs* use a vocabulary borrowed from natural genetics, so the structures or decision rules in a population are called *strings* or *chromosomes*. In our game context, a set of strings would be interpreted as a set of strategies or optimal plans. The performance of the strategies or the decision rules in a given environment is evaluated through their *fitness* functions. In economic modeling, the fitness function measures the value of profit or utility resulting from the behavior prescribed by a given rule or rules. The rules are

⁴ *GA* was developed by Holland (1975).

updated using a set of genetic operators which include *reproduction*, *crossover*, *mutation*, and *election*.

Reproduction makes copies of individual chromosomes. The criterion used in copying is the value of the fitness function, an artificial version of natural selection, a survival of the fittest by Spencer among string creatures. In a natural population, fitness is determined by a creature's ability to survive and reproduce.

The primary genetic operator for *GA* is the crossover operator. The crossover operator is executed in three steps: (1) a pair of strings is chosen from the set of copies; (2) the strings are placed side by side and a point is randomly chosen somewhere along the length of the strings; (3) the segments to the left of the point are exchanged between the strings. For example, a crossover of 111000 and 010101 after the second position produces the offsprings 011000 and 110101. Crossover, working with reproduction according to performance, turns out to be a powerful way of biasing the system towards certain patterns.

Mutation is a secondary search operator which increases the variability of the population. After selection of one individual, each bit position (allele in the chromosome) in the new population undergoes a random change with a probability equal to the mutation rate. For example, after mutation individual 111000 becomes 101000 since the second bit position undergoes a change. A high level of mutation yields an essentially random search.

Election tests the newly generated offsprings before they are permitted to become members of a new population. The *potential* fitness of an offspring is compared to the *actual* fitness values of its parents. Parents are the pairs of strings that are taken from the mating pool for the crossover application. A pair is randomly matched and mated; the parents and children or offsprings form the new population after election.

The algorithm starts by selecting a random sample of M strings (M decision rules), and then applying four operators sequentially. After a new population is created via the mating operator, the algorithm applies the same four operators again, continuing either for a prespecified number of rounds or until a stable population of string values or decision rules emerges. As the "solution" we select the fittest member from the final population.

The reproduction operator increases the representation of relatively fit individuals in the population, but does nothing to find a *fitter* individual. The mutation and mating operator (crossover operator) can add new elements to the population, while destroying old ones. If mutation is applied too frequently (mutation probability is too high), it slows or prevents convergence and degrades the performance of the algorithm because it destroys the fit individuals along with the unfit. The mating operator seems to be a very good device for probabilistically injecting diversity, while giving structures that have proved their fitness a shot at surviving.

This algorithm has proved its value in a variety of applications (Sargent 1993). It has some features of a parallel algorithm, both in the obvious sense that it simultaneously processes a sample distribution of elements, and in the subtler sense that instead of processing individuals, it is really processing equivalence classes of individuals. These equivalence classes, which Holland calls *schemata*, are defined by the lengths of common segments of bit

strings. The algorithm is evidently a random search algorithm, one that does not confine its searches locally⁵.

4 Algorithm For Searching Open-Loop Non-cooperative Solutions

GAs aim at complex problems. They belong to the class of probabilistic algorithms, yet are different from random algorithms as they combine elements of stochastic search. Such genetic-based search methods maintain a population of potential solutions, whereas all other methods process a single point of search space. Thus, a *GA* performs a multi-directional search by maintaining a population of potential solutions and encouraging information formation and exchanges between these directions. In the game-theoretic framework, we use both the optimization and the learning property of the *GA*. At each generation or step, players play the whole game and the scores are rated.

For presentational simplicity, we will restrict attention here to two-player games, but the algorithm can be generalized to N players, each with a different objective function. Thus, in this environment, there are two artificially intelligent players who update their strategies through *GA*, and a fictive player who has full knowledge of both players' actions. The term "fictive player", or "referee", is not essential but we need an intermediary for the exchange of best responses of each player to the action of other players in each generation⁶. This player has no decisive role but provides the best strategies in each iteration to the requested parties synchronically.

In this environment, we have two separate *GAs*. Each *GA* is used to play one side of the problem, and each side has its own evaluation function (utility function, profit function or cost function) and population. The evaluation functions of each player have different parameters and functional form depending on the problem of that player. The crucial part in this algorithm is that the problems of each player are solved *synchronously*. Since we have two different players with different objectives, the problem complexity of each player varies. Thus, based on the complexity of the fitness function, one player might evaluate the performance of his strategies faster than the other player. However, in order to learn the action of the other player against his strategies, each player waits for the other player's action in each generation. Thus the game must be played synchronously and genetic operators must be applied sequentially to each generation (see Fig. 1).

Another crucial part is that the best response of each player is available immediately, and each player decides his own strategies according to the best strategies of the other player. This is the learning process. The information of the best strategies is kept in the *shared memory*, which is

⁵ For application of genetic algorithm to economic problems, see Marimon *et al.* 1990, Marks 1992, Arifovic 1994a, Arifovic 1994b.

⁶ In the *UNIX* system, in order to reach and distribute the information available we used *shared memory*, and call this memory a "fictive player".

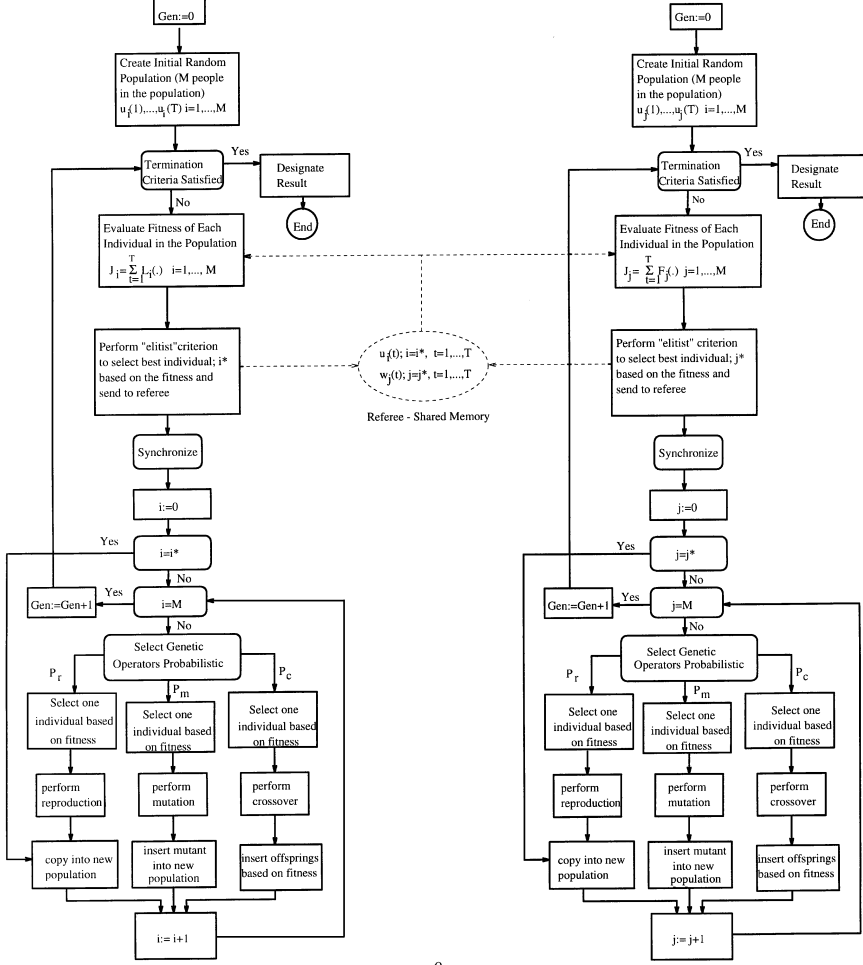


Fig. 1. Flow chart of algorithm for searching open-loop noncooperative

controlled by the fictive player. Each player solves his problem and sends the information about the best decision rules or solutions to the shared memory. Then each player copies the result of the other player's solution and solves his problem again through *GA*.

Initial decision rules are generated randomly for the entire planning period, and in order to measure the performance of these randomly generated rules, each player needs the actions of the other player. Hence, each player sends the very first decision rules generated randomly to the shared memory and the performance of the randomly-generated rules are evaluated. Since *GA* maintains a constant-size population of candidate solutions, we have initially M solutions for each time period. For example, for the two-period problem, we have two decision rules to find, hence initially 2 times M solutions are generated for each player to start the game.

5 Numerical examples

5.1 A simple example

Usually, numerical optimizations and simulations constitute an experiment and, consequently, they should be performed and evaluated with some sort of critical eye appropriate to a laboratory or field experiment. We will offer technical information to the interested researchers for the application of the algorithm for other discrete dynamic games⁷.

A simple numeric example is taken from Kydland (1975). Assume that the problems of two noncooperative players are

$$\max \sum_{t=1}^2 (1 - x_{1t} - x_{2t})x_{it} - \frac{1}{2}u_{it}^2$$

subject to

$$\begin{aligned} x_{it} &= x_{i,t-1} + u_{it} & i &= 1, 2 \\ x_0 &\text{ given} \end{aligned}$$

The maximization is over decision rules; u_{i1} and u_{i2} for $i = 1, 2$. The open-loop decision rules for player 1 are (since the two problems are symmetric, the decision rules are symmetric)

$$\begin{aligned} u_{11} &= -0.6947x_{10} - 0.0947x_{20} + 0.2632 \\ u_{12} &= -0.1789x_{10} + 0.0211x_{20} + 0.0526 \end{aligned}$$

If the initial state variables were $x_{10} = x_{20} = 0.1$, then the open-loop solutions for player 1 would be $u_{11} = 0.1842$ and $u_{12} = 0.0368$.

The algorithm developed in this study starts with the determination of the *GA* parameters which affect the convergence property of the algorithm. However, since the aim of this study is to obtain a general-purpose algorithm for *N*-person difference game, we tried to use parameters which are generalized.

5.1.1 The space of genetic algorithm

Holland (Grefenstette, 1986) describes a fairly general framework for the class of *GAs*. There are many possible elaborations of *GAs* involving variations such as other genetic operators, variable-sized populations etc. This study is limited to a particular subclass of *GAs* characterized by the following parameters: *population size* (M), *crossover rate* (p_c), *mutation rate* (p_m), *generation gap* (G), *selection strategy* (S). Population size affects both performance and efficiency of *GAs*, and a large population is more likely to contain representatives from a large number of hyperplanes. Crossover-rate controls the frequency with which the crossover operation is applied. Mutation, which is the secondary search operator, increases the variability

⁷ Upon request, I will provide the C code of the algorithm by e-mail. My e-mail address is suheyila@bilkent.edu.tr.

of the search spaces. The generation gap controls the percentage of the population to be replaced during each generation. The selection strategy is the *elitist strategy*, which stipulates that the structure with the best performance always survives intact into the next generation. In the absence of such a strategy, it is possible for the best structure to disappear due to sampling error, crossover and mutation.

Here we use parameters $GA = GA(50, 0.6, 0.03, 1.0, E)$. The parameters are population size, crossover rate, mutation rate, generation gap and selection strategy respectively. For an initial population which is generated randomly, player 1 uses random seed 123456789 and player 2 uses 987654321.

5.1.2 Experiment and results

The example given in Kydland is of the two-period, two-player non-cooperative game. Hence, we have to find two optimal strategies, u_{i1} and u_{i2} ($i = 1, 2$), for each player which maximizes the fitness functions of the two players. The fitness function, which measures the performance of each individual strategy in the population f_i , is obtained by substituting constraints into the objective function:

Player 1:

$$f_1 = (1 - x_{10} - u_{11} - x_{20} - u_{21})(x_{10} + u_{11}) - \frac{1}{2}u_{11}^2 + \\ (1 - x_{10} - u_{11} - u_{12} - x_{20} - u_{21} - u_{22})(x_{10} + u_{11} + u_{12}) - \frac{1}{2}u_{12}^2$$

Player 2:

$$f_2 = (1 - x_{10} - u_{11} - x_{20} - u_{21})(x_{20} + u_{21}) - \frac{1}{2}u_{21}^2 + \\ (1 - x_{10} - u_{11} - u_{12} - x_{20} - u_{21} - u_{22})(x_{20} + u_{21} + u_{22}) - \frac{1}{2}u_{22}^2$$

It is worth noting that GA uses the original problem, *not* the first-order conditions for decision-making. As mentioned before, the game starts with the generation of 50 decision rules, u_{i1}, u_{i2} for each player ($i = 1, 2$). For the evaluation of these decisions, each player needs to know the other player's decisions for two periods. Hence, when the random numbers are generated as the potential decision rules of each player, the very first individual decision set in the population is immediately sent to the shared memory by each player and is used as the best decisions of that player for that generation:

Gen	Individual	u_{11}	u_{12}	u_{21}	u_{22}
0	1	0.341924	0.400504	0.285466	0.057359

The fitness functions for generation 1 are calculated for the whole population by using these values as the best decision rules. From Table 1, we can

Table 1. Convergence to the optimal decisions

Gen	u_{11}	u_{12}	f_1	u_{21}	u_{22}	f_2
0	0.055772	0.094682	0.225860	0.105699	0.015076	0.242384
1	0.177234	0.044419	0.221428	0.194690	0.012314	0.221126
5	0.177234	0.044419	0.218435	0.188846	0.032502	0.221697
10	0.177127	0.043458	0.218440	0.188846	0.032502	0.222071
15	0.179187	0.044419	0.214624	0.189212	0.043656	0.220319
20	0.179187	0.044312	0.214872	0.188846	0.043580	0.220370
25	0.179080	0.040299	0.218371	0.188891	0.032761	0.221896
30	0.179080	0.040299	0.218371	0.188891	0.032761	0.221896
35	0.183627	0.041459	0.220636	0.184390	0.034211	0.218788
40	0.183627	0.039750	0.220649	0.184390	0.034211	0.219333
45	0.183627	0.039750	0.220358	0.184390	0.035111	0.219336
50	0.183627	0.039750	0.220518	0.184085	0.035187	0.219336
200	0.184115	0.037003	0.219864	0.184375	0.036698	0.219923
400	0.184207	0.036835	0.219916	0.184207	0.036851	0.219921
600	0.184207	0.036835	0.219912	0.184222	0.036835	0.219921
800	0.184207	0.036851	0.219912	0.184222	0.036835	0.219916
1000	0.184207	0.036851	0.219921	0.184207	0.036835	0.219916
1200	0.184207	0.036851	0.219916	0.184207	0.036851	0.219916
1400	0.184207	0.036851	0.219916	0.184207	0.036851	0.219916
1495	0.184207	0.036851	0.219916	0.184207	0.036851	0.219916

follow the generation of new individuals and their convergence to the optimal decisions for each player:

At trial 50000, the open-loop decision rules for player 1 and player 2 have almost been reached by *GA*. However, we ran 25000 more trials, to decide whether *GA* converged to the optimal rules. When compared to the exact solutions, it is apparent that *GA* works, the theory being that an optimally intelligent system should process currently available information about payoffs from the unknown environment so as to find the optimal tradeoff between the cost of *exploration* of new points in the search space and the cost of *exploitation* of already evaluated points in the search space.

5.2 A wage bargaining game

A typical example of policy optimization in a policy game is given by Brandsma and Hughes Hallett (1984) using the formulation by L.R. Klein to describe the mechanism of the American economy in the interwar period, 1933-1936. Theil (1964) reduced Klein's highly aggregative equation system into three instruments and three noncontrolled (target) variables. The targets (consumption C , investment I , and income distribution D) are linked to three instruments (public sector wages W_2 , the indirect tax yield T , and government expenditures G). The decision problem of Roosevelt is to resolve the depression and to return economic activity per capita to at least its previous peak (1929) level. Brandsma and Hughes Hallett partition the available instruments as W_2 for one player (organized labor) and T and G for the second player (government or budgetary authority). The idea is to

examine the optimal policies in a noncooperative game between the Roosevelt administration and organized labor as the economic strategy of the New Deal was introduced to counter the great recession⁸. In Theil's exercise, current welfare is loosely represented by consumption, future welfare by investment, and the distribution of income between capital and labor reflects the New Deal commitment to organized labor.

From the above, desired consumption in 1936 is $C_{36}^d = C_{29}(1 + \alpha)^7$ where $\alpha = 0.1$ is the observed population growth rate and the subscripts denote the year ($C_{29} = 57.8$ in billions of dollars of 1934, is the realized level of total consumption in 1929). As to investment, I , it appears that this variable was of the order of 10 percent of consumption during the 1920s, and hence, desired investment in 1939 averaged $I_{36}^d = 0.1C_{36}^d$. Theil took $D = W_1 - 2\pi$ as the income distribution, where W_1 is private wage bill and π is profit, and put its desired level in 1936 equal to zero ($D_{36}^d = 0$). To specify desired values for the intermediate years, 1933–1935, simple linear interpolation is done between the actual values in 1932 (the last year before the Roosevelt administration) and the corresponding desired values in 1936. Hence, $C_{32+t}^d = C_{32} + \frac{t}{4}(C_{36}^d - C_{32})$ and $I_{32+t}^d = I_{32} + \frac{t}{4}(C_{36}^d - C_{32})$ for $t = 1, 2, 3$. Since $D_{36}^d = 0$, $D_{33}^d, \dots, D_{35}^d$ is fixed by interpolation.

A game naturally arises here through conflicting private interests of the government (or employer) and organized labor in rebuilding consumption and investment. Organized labor would press for faster increases in the wage bill (W_1) in order to restore their previous earning and employment levels. But employers would demand a restoration of profits (by 1932, D was large and positive) in order to boost investment, and the government would have to submit in order to secure the welfare. This implies the employees would have ideal values for D , which remain above Theil's smooth restoration of status quo ante, while the employers and government would set values that imply a faster return to zero, demonstrating that the administration was fully aware of this conflict and the need to resolve it.

The desired level of the instruments is handled in an analogous manner. Following the trend argument, the desired values are projections of the observed trends in the associated variables over the period 1920–32. The numerical specification of these values are given in Table 2 (Brandsma and Hughes Hallett, 1984).

The objective function was formulated as the minimization of the sum of squares of the deviations between actual and desired values of variables which are the interests of each player. Thus, the private interests of each player i , can be represented by the quadratic loss function,

$$w^{(i)} = \frac{1}{2} (\tilde{y}^{(i)'} B^{(i)} \tilde{y}^{(i)} + \tilde{x}^{(i)'} A^{(i)} \tilde{x}^{(i)}), \quad i = 1, 2$$

⁸ Organized labor was one of the chief groups in the coalition which brought Roosevelt to power. It had suffered particularly from the recession and unemployment after 1929, so it could have been expected to attempt to extract a price for its continued support. Its instruments would be power to influence wage demands and hence income distribution as well as the industrial relations in public and private sector (Hughes Hallett and Rees, 1983).

Table 2. Desired Values of Instruments and Target Variables

Policy	1933	1934	1935	1936
Labor				
C	49.49	53.78	57.88	61.97
I	-3.10	0.00	3.10	6.20
D	12.50	10.00	7.50	5.00
W ₂	5.04	5.25	5.47	5.69
Government				
C	49.49	53.78	57.88	61.97
I	-3.10	0.00	3.10	6.20
D	0.00	0.00	0.00	0.00
T	7.40	7.64	7.87	8.11
G	10.44	10.87	11.30	11.73

where $\tilde{y}^{(i)} = y^{(i)} - y^{(i)d}$ is the vector of target variables, $\tilde{x}^{(i)} = x^{(i)} - x^{(i)d}$ is the vector of instruments, and $B^{(i)}$ and $A^{(i)}$ are positive definite symmetric matrices. These matrices, $B^{(i)}$, $A^{(i)}$ denote penalties, which implies that accelerating unit penalties accrue to persistent or cumulating failures in electorally significant variables. The private objective functions were specified by picking out the relevant penalties from those revealed in the historical policy decisions (Ancot et al., 1982). The objective function is summarized in Table 3.

Table 3. The Preference Structure: penalties on squared failures

Policy	1933	1934	1935	1936	Preference Matrix
Labor					
C	1.0	1.0	1.0	1.0	B
I	0.01	0.3	1.0	0.5	B
D	0.5	0.5	2.0	5.0	B
W ₂	1.0	2.0	1.0	5.0	A
Government					
C	1.0	2.0	5.0	4.0	B
I	0.1	0.3	1.0	0.5	B
D	0.5	0.4	2.0	0.25	B
T	1.0	0.1	0.1	0.8	A
G	2.0	1.0	1.0	1.0	A
T and G	0.25	0.25	0.0	-0.4	A
C and I	0.05	0.0	1.0	0.0	B
C and D	0.3	0.0	-1.0	-1.0	B
Intertemporal Penalties:					
T ₃₅ and T ₃₆ :	-0.1	T ₃₅ and G ₃₆ :	-0.1	G ₃₃ and T ₃₆ :	-0.5
G ₃₃ and G ₃₆ :	0.5	G ₃₅ and T ₃₆ :	-0.2	G ₃₅ and G ₃₆ :	0.3
C ₃₃ and C ₃₅ :	0.5	C ₃₃ and C ₃₆ :	0.5	C ₃₅ and C ₃₆ :	2.0
D ₃₅ and C ₃₆ :	-1.0	C ₃₃ and G ₃₆ :	0.5		

Player 1 has one instrument $x^{(1)} = W_2$, while player 2 has two instruments $x^{(2)} = T, G$. Also, each decision maker has ideal values $y^{(i)d}, x^{(i)d}$ for his own decision variables, so that $\tilde{y}^{(i)} = y^{(i)} - y^{(i)d}$ and $\tilde{x}^{(i)} = x^{(i)} - x^{(i)d}$ define his policy failures. Thus each player's aim is to minimize his failures subject to constraints, implied by the econometric model:

$$y^j = R^{i,1}x^1 + R^{i,2}x^2 + s^i, \quad i = 1, 2$$

where $R^{i,j}, j = 1, 2$ are known matrices containing the dynamic multipliers $R_{tk}^{i,j} = \partial y_t^i / \partial x_k^j$ if $t \geq k$ and zeros otherwise, and where s^i is the subvector of s associated with y^i and denotes structural disturbances. Thus $R^{i,i}$ describes the response of player i 's targets to his own instruments, and $R^{i,j}$ their responses to player j 's decisions. Since each player has the same noncontrollable variables, the constraint equation can be written in aggregate form:

$$y = Rx + s$$

or, more exclusively, as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} R_1 & & & \\ R_2 & R_1 & & \\ R_3 & R_2 & R_1 & \\ R_4 & R_3 & R_2 & R_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

where $x' = (x^{(1)'}, x^{(2)'})$ is the vector of instrument variables and the coefficient matrices R_1 measure the effectiveness of instruments with respect to current uncontrolled variables; hence they are equal to submatrices $R_{11}^{(i)}, R_{22}^{(i)}, R_{33}^{(i)}, R_{44}^{(i)}$. In the same way, the matrix R_2 measures the effectiveness of the instruments with respect to uncontrolled variables one year later, and is equal to submatrices $R_{21}^{(i)}, R_{32}^{(i)}, R_{43}^{(i)}$. And so on. The submatrices of the multiplicative and the additive structure of the constraint are summarized as in Table 4 (Theil, 1964).

Table 4. The Multiplicative and Additive Structure of the Constraints

	W_2	T	G	
		R_1		s_1
C	0.666	-0.188	0.671	37.55
I	-0.052	-0.296	0.259	-5.62
D	0.285	2.358	1.427	5.04
		R_2		s_2
C	-0.234	-1.014	1.170	35.18
I	-0.152	-0.894	0.759	-4.64
D	0.095	1.172	-0.475	0.08
		R_3		s_3
C	-0.172	-1.006	0.859	36.55
I	-0.076	-0.518	0.382	-2.49
D	-0.007	0.186	0.033	-3.61
		R_4		s_4
C	-0.079	-0.543	0.396	42.84
I	-0.005	-0.088	0.024	2.11
D	-0.060	-0.285	0.301	-11.91

Evidently, each player's optimal strategy depends on and must be determined simultaneously with the optimal decisions to be expected from the other player. In the absence of cooperation, the optimal decisions $(x^{(1)*}, x^{(2)*})$ will satisfy

$$w^{(i)}(x^{(i)*}, x^{(j)*}) \leq w^{(i)}(x^{(i)}, x^{(j)*})$$

for $i = 1, 2; i \neq j$ for all feasible $x^{(i)} \neq x^{(i)*}$.

5.2.1 Optimal strategies

The numerical result of this wage-setting game is summarized in Table 5.

Table 5. Open-Loop Nash Strategies in 1934 billions dollar

Policy	1933	1934	1935	1936
Instruments:				
W_2	5.621	6.591	11.505	7.621
T	5.139	6.099	4.805	8.122
G	11.690	10.929	10.001	11.365
Target Strategies:				
C	48.172	52.908	58.985	61.666
I	-4.406	-0.533	1.297	4.115
D	2.077	1.748	0.612	-1.665
$\omega^{(1)} = 237.187$				
$\omega^{(2)} = 10.765$				

Before examining the result, the technical points can be summarized as follows: population size is 50, crossover rate is 0.6, mutation rate is 0.03 and finally, number of trials is 2 million. The optimal strategies are tested to see whether they satisfy necessary conditions (first and second order conditions).

Since GA does not use first-order conditions for decision-making, we have a chance to test whether the results work. Then the necessary conditions for an optimal strategy to hold is the following:

$$\partial w^{(i)} / \partial x^{(i)} + (\partial y^{(i)} / \partial x^{(i)})' \partial w^{(i)} / \partial y^{(i)} = 0$$

for all the estimated decision rules $x^{(i)}$'s. The result showed that the first order condition hold with 0.003 error. Since this is a numeric study, the results are all near optimal and quite convincing.

The results show rather stable values for the government expenditure on goods and services. The values are consistently above their desired levels, which is, of course, in accordance with generally accepted ideas about anti-depression policies. Regarding the uncontrollable variables, C and I , we observe that strategy values are below the desired levels, which is also in accordance with a depression situation.

5.3 Why does GA work?

One method of obtaining the open-loop Nash equilibrium solutions of the class of discrete time games is to view them as static infinite games, and

directly apply the methodology that minimizes cost functional over control sets and then determines the intersection points of the resulting reaction curves. Such an approach can sometimes lead to quite unwieldy expressions, especially if the number of stages in the game is large (Başar and Oldser, 1982). An alternative derivation, which partly removes this difficulty, is the one that utilizes techniques of optimal control theory, by making explicit use of the stage additive nature of the cost functionals and the specific structure of the extensive-form description of the game (For a more general treatment see Başar and Oldser, chapter 6, 1982). There is, in fact, a close relationship between the derivation of open-loop Nash equilibria and the problem of solving (jointly) N optimal control problems, since each N -tuple of strategies constituting a Nash equilibrium solution describes an optimal control problem whose *structure* is not affected by the remaining players' control vectors. This structure of dynamic or differential games enables *GA* to work in deriving optimal open-loop strategies.

Amenability to parallelization is an appealing feature of the conventional genetic algorithm. In genetic methods, the genetic operations themselves are very simple and not very time-consuming; whereas measurement of fitness of the individuals in the population is typically complicated and time-consuming. In considering approaches for parallelizing genetic methods, it is important to note that, for all but the most trivial problems, the majority of the computational effort is consumed by the calculation of fitness. The execution of the genetic operation can be parallelized in the manner described by Robertson (1987); however, we focus here on ways of parallelizing genetic programming that distribute the computational effort needed to compute fitness. There are two basic approaches to parallelization. In one of the approaches, the *distributed genetic approach*, the population for a given run is divided into subpopulations (Koza, 1993). Each subpopulation is assigned to a processor, and the genetic algorithm operates on each separately. Upon completion of a certain designated number of generations, a certain percentage of the individuals in each population are selected for emigration, and there is a partial exchange of members between the subpopulations.

The algorithm developed in this paper follows a similar idea while exchanging the best results in each generation. The main goal in this study is to show the success of the *algorithm* developed here in solving nonlinear dynamic games with some empirical evidence.⁹ For problems that violate the appropriate dimensionals, we apply numerical methods.

Future research might profitably examine the closed-loop non-cooperative solutions. For closed-loop solution procedure, we need to work on the objective function of each player separately to make selection. So, in this case, the most important part will be to devise a more efficient *selection* procedure. In our open-loop procedure, we were selecting new populations at the same time, but in the closed-loop, we might offer intersections or unions of the selections done separately to form new populations and apply

⁹ Başar and Oldser proved that an N -person linear quadratic dynamic game with appropriate dimensional matrices, has a unique open-loop Nash equilibrium solution.

other genetic operators. Each individual's gene will be attached to another (*concatanation*) of the genes in the shared memory algorithm. So, to use *GA* to solve closed-loop problems we have to write a new selection procedure such as: Initially using the first players objective function, we make a selection and form a set of solutions of that player, and by doing the same procedure, we form the other palyer's set of solutions. By either an intersection or union of these sets, we form a new set of solutions. However, the critical problem in the solution procedure for the closed-loop games will be selecting the new population set. We have to further study whether intersection or union will be the suitable selection procedure.

References

- Ancot JP, Hughes Hallett AJ, Paelinck JHP (1982) The determination of implicit preferences: Two possible approaches compared. *European Economic Review* 18: 267–289
- Arifovic J (1994) Genetic algorithm learning and the cobweb model. *Journal of Economic Dynamics and Control* 18: 3–28
- Arifovic J (1994) The behavior of the exchange rate in genetic algorithm and experimental economics. Simon Fraser University working papers
- Başar T, Oldser GJ (1982) *Dynamic noncooperative game theory*. Academic Press New York
- Bona JL, Santos MS (1994) On the role of computation in economic theory. Universidad Carlos III working paper
- Brandsma AS, Hughes Hallett AJ (1984) Economic conflict and the solution of dynamic games. *European Economic Review* 26: 13–32
- De Bruyne G (1979) Pareto optimality of noncooperative equilibrium in a time dependent multiperiod game. *European Economic Review* 12: 243–260
- De Zeeuw AJ, Van der Ploeg F (1991) Difference games and policy evaluation: A conceptual framework. *Oxford Economic Papers* 43: 612–636
- Goldberg DE (1989) *Genetic algorithm in search, optimization, and machine learning*. Addison-Wesley, Reading Massachusetts
- Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics* 16: 122–128
- Ho YC (1970) Differential games, dynamic optimization, and generalized control theory. *Journal of Optimization Theory and Applications* 6: 179–209
- Hughes-Hallett AJ, Rees HJB (1983) *Quantitative economic policies and interactive planning*. Cambridge University Press, Cambridge New York
- Judd KL (1992) Projection methods for solving aggregate growth models. *Journal of Economic Theory* 58: 410–452
- Karp LS, McCalla AF (1983) Dynamic games and international trade: An application to the world corn market. *American Journal of Agricultural Economics* 65: 641–650
- Kendrick D, Taylor L (1970) Numerical solution on nonlinear planning models. *Econometrica* 38: 453–467
- Koza JR (1992) *Genetic programming*. The MIT Press, Cambridge Massachusetts
- Kydland F (1975) Noncooperative and dominant player solutions in discrete dynamic games. *International Economic Review* 16: 321–335
- Marimon R, McGrattan E, Sargent TJ (1990) Money as a medium of exchange in an economy with artificially intelligent agents. *Journal of Economic Dynamics and Control* 14: 329–373
- Marks RE (1992) Breeding hybrid strategies: optimal behaviour for oligopolists. *Journal of Evolutionary Economics* 2: 17–38
- Miller M, Salmon M (1983) Policy coordination and dynamic games. In Buitter WH and RC Marston (eds) *International Economic Policy Coordination*. Cambridge University Press Cambridge

- Pau LF (1975) A differential game among sectors in a macroeconomy. *Automatica* 11: 473–485
- Pindyck RS (1977) Optimal economic stabilization under decentralized control and conflicting objectives. *IEEE Transactions and Automatic Control* AC-22: 517–530
- Robertson G (1987) Parallel implementation of genetic algorithms in a classifier systems. In Davis L (ed) *Genetic Algorithms and Simulated Annealing*. Pittman London
- Sargent TJ (1993) *Bounded Rationality in Macroeconomics*. Oxford University Press Oxford
- Starr AW, Ho YC (1969) Further Properties of nonzero sum differential games. *Journal of Optimization Theory and Applications* 3: 207–219
- Theil H (1964) *Optimal decision rules for government and industry*. North-Holland Amsterdam
- Van der Ploeg F (1982) Government policy, real wage resistance and the resolution of conflict. *European Economic Review* 19: 181–212